FIG. 1A

LS0 TYPE (BASE RELATIVE ADDRESSING)

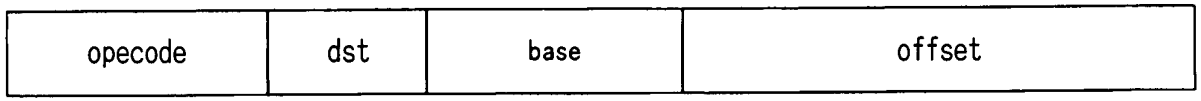


FIG. 1B

LS1 TYPE (POST OFFSET UPDATE ADDRESSING)

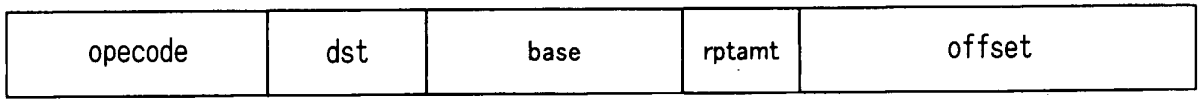


FIG. 1C

LS2 TYPE (INDEX MODIFICATION/POST REGISTER UPDATE ADDRESSING)

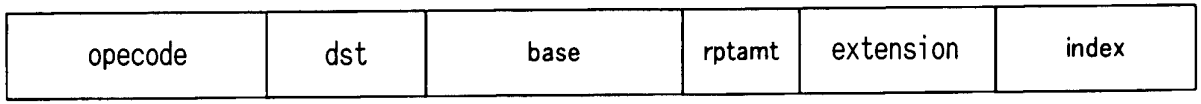


FIG. 2

extension	ADDRESS MODIFICATION METHOD

000	INTERPRETS THE CONTENT OF THE REGISTER SPECIFIED IN THE "INDEX" AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].h[LOWER];
	}

001	INTERPRETS THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[LOWER]) AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].h[LOWER];
	}

010	INTERPRETS THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[UPPER]) AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].h[UPPER];
	}

011	INTERPRETS EACH OF THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[LOWER]) AND THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[UPPER]) AS SIGNED AND ALTERNATELY PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  if ((t & 1)==0) targetAdrs = base.uw + index[t].h[LOWER];
	  else targetAdrs = base.uw + index[t].h[UPPER];
	}

FIG. 3

extension	ADDRESS MODIFICATION METHOD

000	INTERPRETS THE CONTENT OF THE REGISTER SPECIFIED IN THE "INDEX" AS SIGNED AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].w;
	}
	base.uw = T;

001	INTERPRETS THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[LOWER] AS SIGNED AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[LOWER];
	}
	base.uw = T;

010	INTERPRETS THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[UPPER] AS SIGNED AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[UPPER];
	}
	base.uw = T;

011	INTERPRETS EACH OF THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[LOWER], AND THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[UPPER], AS SIGNED AND ALTERNATELY UPDATES THE ADDRESSES.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  if ((t & 1) == 0) T += index[t].h[LOWER];
	  else T += index[t].h[UPPER];
	}
	base.uw = T;

100	INTERPRETS THE CONTENT OF THE REGISTER SPECIFIED IN THE "INDEX" AS SIGNED AND UPDATES THE ADDRESS. BASE REGISTER IS NOT UPDATED.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].w;
	}

101	INTERPRETS THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[LOWER] AS SIGNED, AND UPDATES THE ADDRESS. BASE REGISTER IS NOT UPDATED.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[LOWER];
	}

110	INTERPRETS THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[UPPER] AS SIGNED, AND UPDATES THE ADDRESS. BASE REGISTER IS NOT UPDATED.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[UPPER];
	}

111	INTERPRETS EACH OF THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[LOWER], AND THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" , index[t].h[UPPER], AS SIGNED AND ALTERNATELY UPDATES THE ADDRESSES.
	BASE REGISTER IS NOT UPDATED.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  if ((t & 1) == 0) T += index[t].h[LOWER];
	  else T += index[t].h[UPPER];
	}

FIG. 4

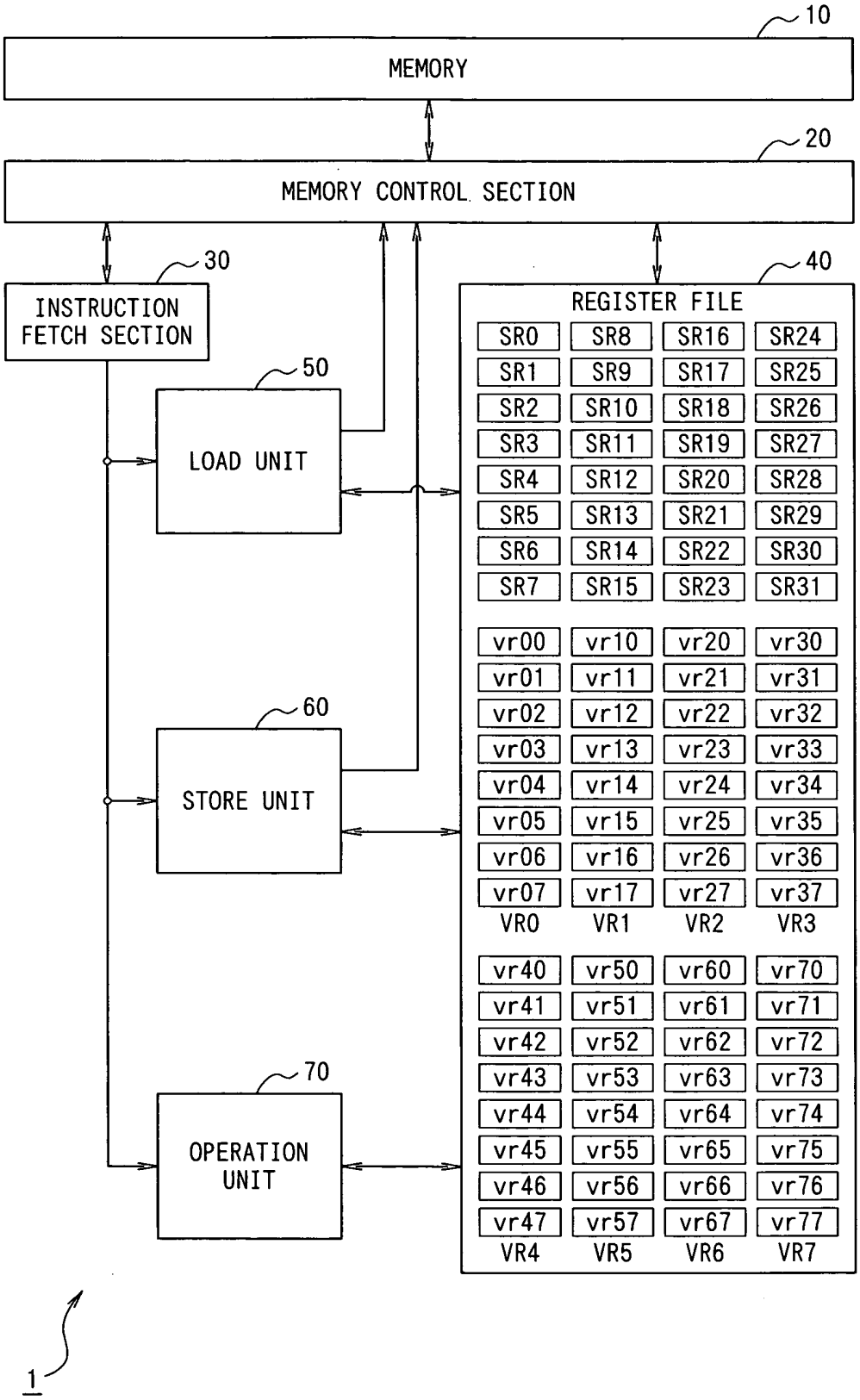

10
MEMORY
20
MEMORY CONTROL SECTION
30
INSTRUCTION FETCH SECTION
40
REGISTER FILE
50
LOAD UNIT
60
STORE UNIT
70
OPERATION UNIT
SR0 SR8 SR16 SR24
SR1 SR9 SR17 SR25
SR2 SR10 SR18 SR26
SR3 SR11 SR19 SR27
SR4 SR12 SR20 SR28
SR5 SR13 SR21 SR29
SR6 SR14 SR22 SR30
SR7 SR15 SR23 SR31
vr00 vr10 vr20 vr30
vr01 vr11 vr21 vr31
vr02 vr12 vr22 vr32
vr03 vr13 vr23 vr33
vr04 vr14 vr24 vr34
vr05 vr15 vr25 vr35
vr06 vr16 vr26 vr36
vr07 vr17 vr27 vr37
VR0 VR1 VR2 VR3
vr40 vr50 vr60 vr70
vr41 vr51 vr61 vr71
vr42 vr52 vr62 vr72
vr43 vr53 vr63 vr73
vr44 vr54 vr64 vr74
vr45 vr55 vr65 vr75
vr46 vr56 vr66 vr76
vr47 vr57 vr67 vr77
VR4 VR5 VR6 VR7
1

FIG. 5

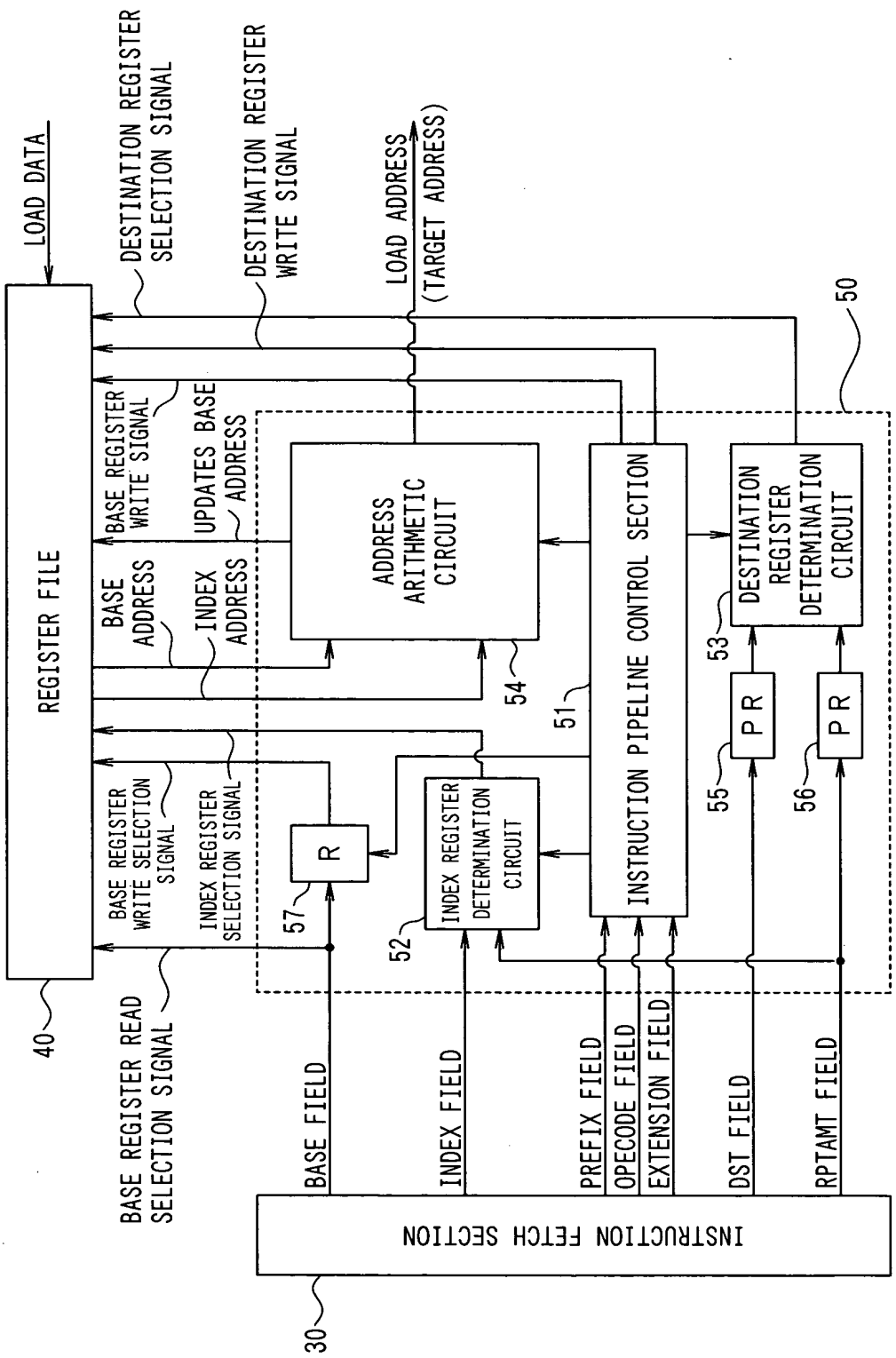

LOAD DATA
DESTINATION REGISTER SELECTION SIGNAL
DESTINATION REGISTER WRITE SIGNAL
LOAD ADDRESS (TARGET ADDRESS)
50
REGISTER FILE
BASE REGISTER WRITE SIGNAL
UPDATES BASE ADDRESS
BASE ADDRESS
INDEX ADDRESS
ADDRESS ARITHMETIC CIRCUIT
INSTRUCTION PIPELINE CONTROL SECTION
DESTINATION REGISTER DETERMINATION CIRCUIT
53
54
51
P R
55
P R
56
BASE REGISTER WRITE SELECTION SIGNAL
INDEX REGISTER SELECTION SIGNAL
R
57
INDEX REGISTER DETERMINATION CIRCUIT
52
40
BASE REGISTER READ SELECTION SIGNAL
BASE FIELD
INDEX FIELD
PREFIX FIELD
OPECODE FIELD
EXTENSION FIELD
DST FIELD
RPTAMT FIELD
INSTRUCTION FETCH SECTION
30

FIG. 6

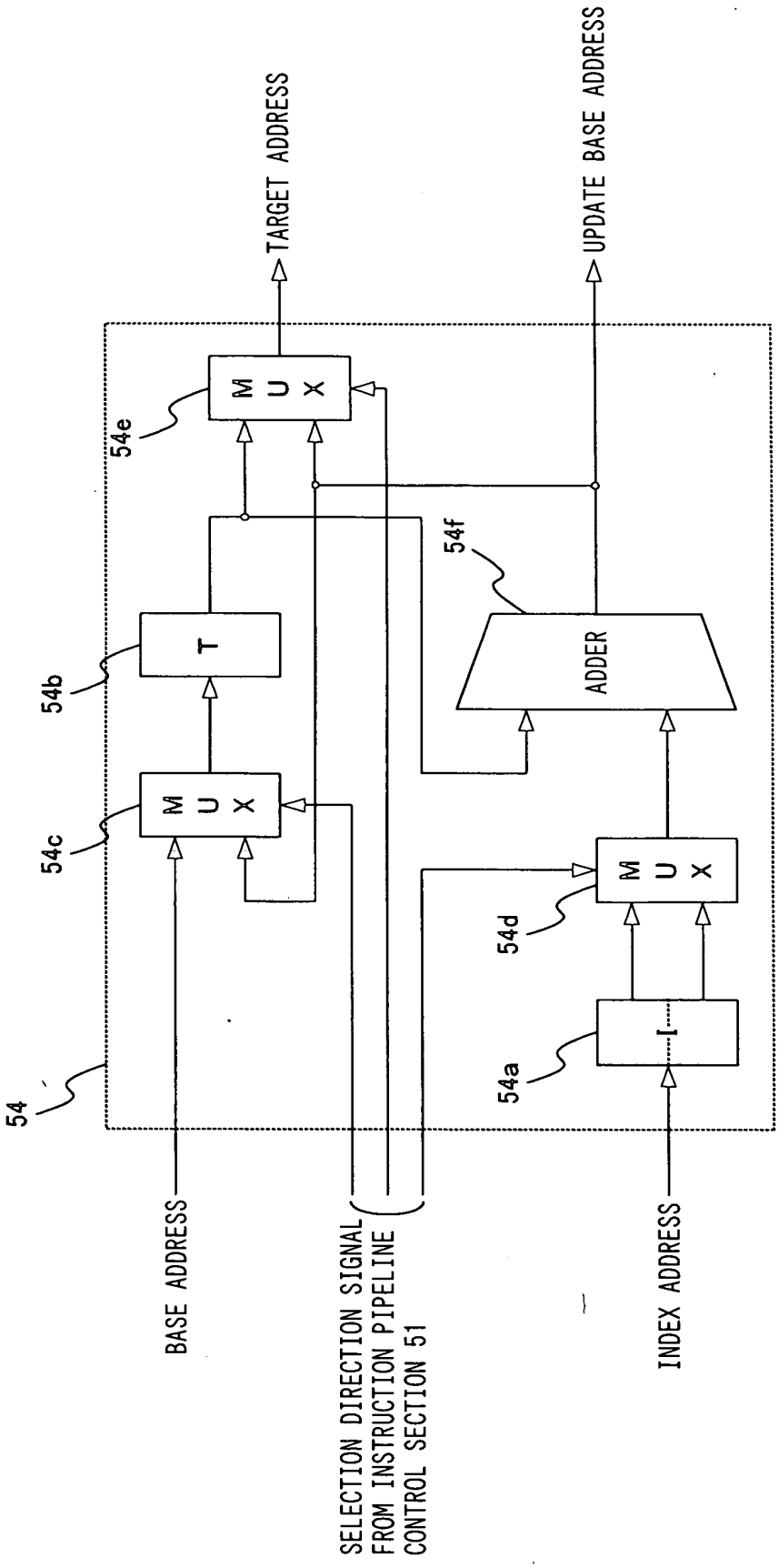

TARGET ADDRESS
UPDATE BASE ADDRESS
54e
MUX
54f
ADDER
54b
T
54c
MUX
54d
MUX
54a
I
54
BASE ADDRESS
SELECTION DIRECTION SIGNAL
FROM INSTRUCTION PIPELINE
CONTROL SECTION 51
INDEX ADDRESS

FIG. 7

extension	ADDRESS MODIFICATION METHOD

0000	INTERPRETS THE CONTENT OF THE REGISTER SPECIFIED IN THE "INDEX" AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].w;
	}

0100	INTERPRETS THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[LOWER]) AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].h[LOWER];
	}

0110	INTERPRETS THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" (index[t].h[UPPER]) AS SIGNED AND PERFORMS ADDRESS MODIFICATION.
	for(t=0; t < rptamt; t++){
	  targetAdrs = base.uw + index[t].h[UPPER];
	}

extension	ADDRESS MODIFICATION METHOD

0000	INTERPRETS THE CONTENT OF THE REGISTER SPECIFIED IN THE "INDEX" AS SIGNED AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].w;
	}
	base.uw = T;

0100	INTERPRETS THE LOWER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" ,
	index[t].h[LOWER] AS SIGNED, AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[LOWER];
	}
	base.uw = T;

0101	THE SAME AS ABOVE. BASE REGISTER IS NOT UPDATED, HOWEVER.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[LOWER];
	}

0110	INTERPRETS THE UPPER 16 BITS OF THE REGISTER SPECIFIED IN THE "INDEX" ,
	index[t].h[UPPER] AS SIGNED, AND UPDATES THE ADDRESS.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[UPPER];
	}
	base.uw = T;

0111	THE SAME AS ABOVE. BASE REGISTER IS NOT UPDATED, HOWEVER.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T; T += index[t].h[UPPER];
	}

1100	INTERPRETS ANY DATA OF THE LOWER 16 BITS index[t].h[LOWER] AND THE UPPER 16 BITS
	index[t].h[UPPER] OF THE REGISTER SPECIFIED IN THE "INDEX" , AND THE LOWER 16 BITS
	SR30.h[LOWER] AND THE UPPER 16 BITS SR30.h[UPPER] OF THE REGISTER A AS SIGNED,
	AND ALTERNATELY UPDATES THE ADDRESSES.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  switch((ISW.h[LOWER] >> (2 * t)) & 0x3){
	  case 0: T += index.h[LOWER]; break;
	  case 1: T += index.h[UPPER]; break;
	  case 2: T +=SR30.h[LOWER]; break;
	  case 3: T +=SR30.h[UPPER]; break;
	  }
	}
	base.uw = T;

1101	THE SAME AS ABOVE. BASE REGISTER IS NOT UPDATED, HOWEVER.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  switch((ISW.h[UPPER] >> (2 * t)) & 0x3){
	  case 0: T += index.h[LOWER]; break;
	  case 1: T += index.h[UPPER]; break;
	  case 2: T +=SR30.h[LOWER]; break;
	  case 3: T +=SR30.h[UPPER]; break;
	  }
	}

1110	INTERPRETS ANY DATA OF THE LOWER 16 BITS index[t].h[LOWER] AND THE UPPER 16 BITS
	index[t].h[UPPER] OF THE REGISTER SPECIFIED IN THE "INDEX" , AND THE LOWER 16 BITS
	SR31.h[LOWER] AND THE UPPER 16 BITS SR31.h[UPPER] OF THE REGISTER B AS SIGNED
	AND ALTERNATELY UPDATES THE ADDRESSES.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  switch((ISW.h[LOWER] >> (2 * t)) & 0x3){
	  case 0: T += index.h[LOWER]; break;
	  case 1: T += index.h[UPPER]; break;
	  case 2: T +=SR31.h[LOWER]; break;
	  case 3: T +=SR31.h[UPPER]; break;
	  }
	}
	base.uw = T;

1111	THE SAME AS ABOVE. BASE REGISTER IS NOT UPDATED, HOWEVER.
	T = base.uw;
	for(t=0; t < rptamt; t++){
	  targetAdrs = T;
	  switch((ISW.h[UPPER] >> (2 * t)) & 0x3){
	  case 0: T += index.h[LOWER]; break;
	  case 1: T += index.h[UPPER]; break;
	  case 2: T +=SR31.h[LOWER]; break;
	  case 3: T +=SR31.h[UPPER]; break;
	  }
	}

FIG. 9

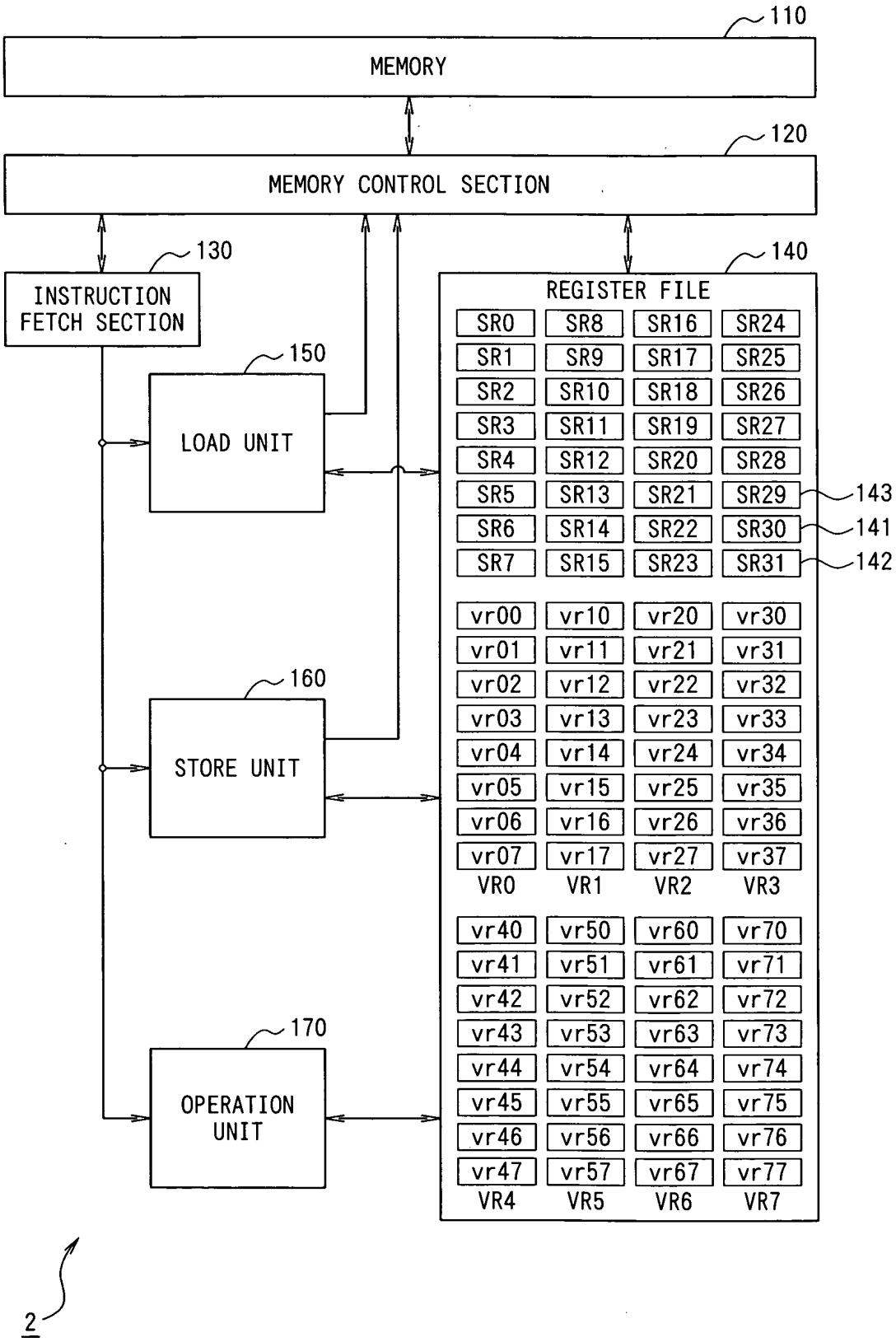

110
MEMORY
120
MEMORY CONTROL SECTION
130
INSTRUCTION FETCH SECTION
140
REGISTER FILE
150
LOAD UNIT
160
STORE UNIT
170
OPERATION UNIT
SR0 SR8 SR16 SR24
SR1 SR9 SR17 SR25
SR2 SR10 SR18 SR26
SR3 SR11 SR19 SR27
SR4 SR12 SR20 SR28
SR5 SR13 SR21 SR29
SR6 SR14 SR22 SR30
SR7 SR15 SR23 SR31
143
141
142
vr00 vr10 vr20 vr30
vr01 vr11 vr21 vr31
vr02 vr12 vr22 vr32
vr03 vr13 vr23 vr33
vr04 vr14 vr24 vr34
vr05 vr15 vr25 vr35
vr06 vr16 vr26 vr36
vr07 vr17 vr27 vr37
VR0 VR1 VR2 VR3
vr40 vr50 vr60 vr70
vr41 vr51 vr61 vr71
vr42 vr52 vr62 vr72
vr43 vr53 vr63 vr73
vr44 vr54 vr64 vr74
vr45 vr55 vr65 vr75
vr46 vr56 vr66 vr76
vr47 vr57 vr67 vr77
VR4 VR5 VR6 VR7
2

FIG. 10

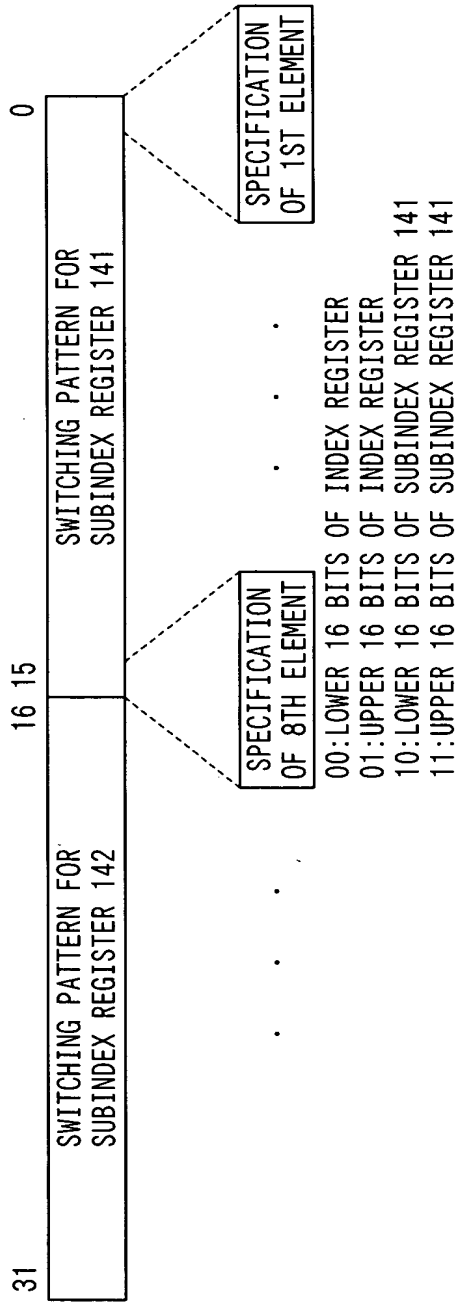

31
16 15
0
SWITCHING PATTERN FOR SUBINDEX REGISTER 142
SWITCHING PATTERN FOR SUBINDEX REGISTER 141
SPECIFICATION OF 8TH ELEMENT
SPECIFICATION OF 1ST ELEMENT
00:LOWER 16 BITS OF INDEX REGISTER
01:UPPER 16 BITS OF INDEX REGISTER
10:LOWER 16 BITS OF SUBINDEX REGISTER 141
11:UPPER 16 BITS OF SUBINDEX REGISTER 141

FIG. 11

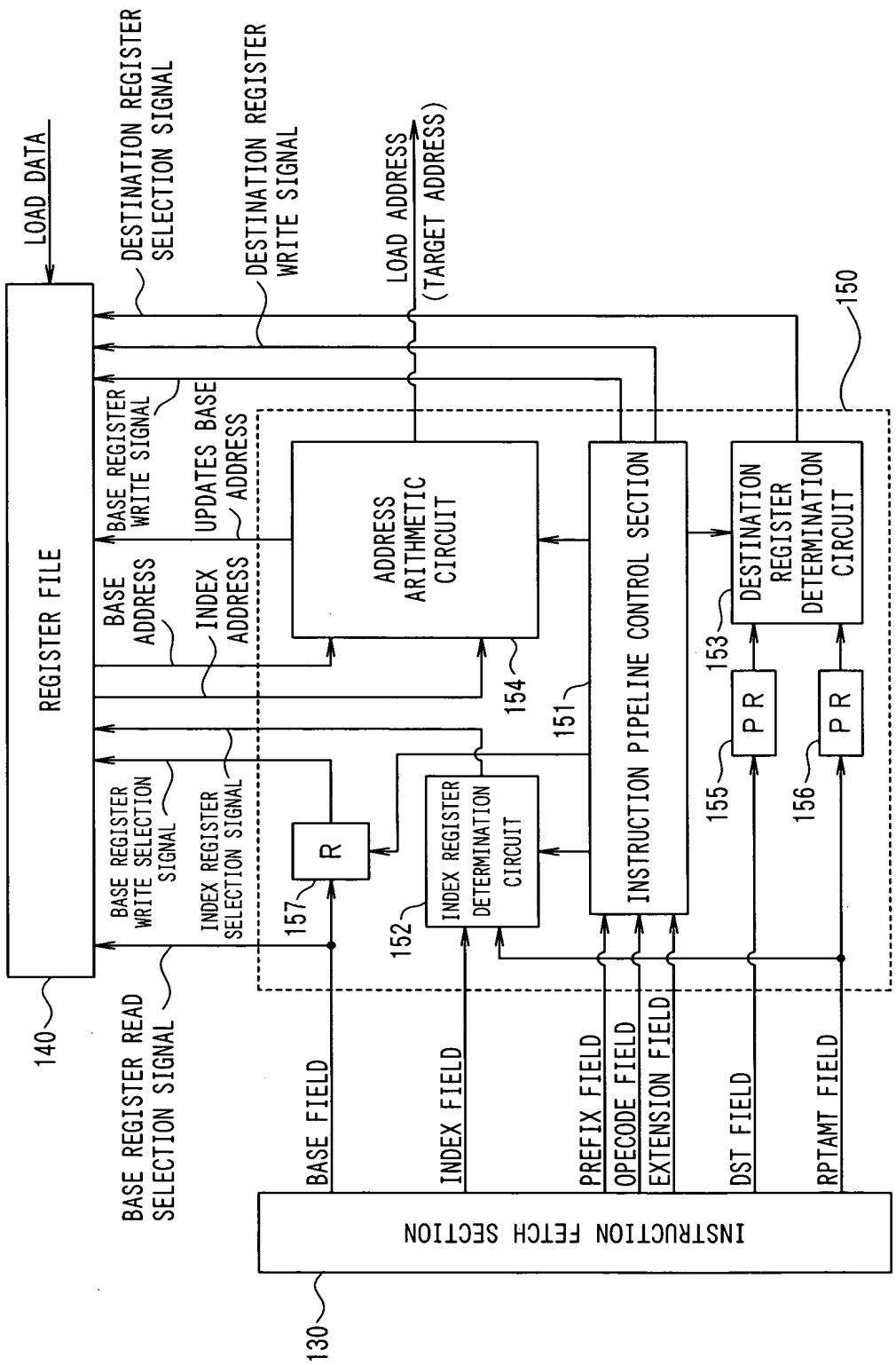

LOAD DATA
DESTINATION REGISTER SELECTION SIGNAL
DESTINATION REGISTER WRITE SIGNAL
LOAD ADDRESS (TARGET ADDRESS)
150
REGISTER FILE
BASE REGISTER WRITE SIGNAL
UPDATES BASE ADDRESS
BASE ADDRESS
INDEX ADDRESS
ADDRESS ARITHMETIC CIRCUIT
INSTRUCTION PIPELINE CONTROL SECTION
DESTINATION REGISTER DETERMINATION CIRCUIT
153
154
151
P R
P R
155
156
BASE REGISTER WRITE SELECTION SIGNAL
INDEX REGISTER SELECTION SIGNAL
R
157
INDEX REGISTER DETERMINATION CIRCUIT
152
140
BASE REGISTER READ SELECTION SIGNAL
BASE FIELD
INDEX FIELD
PREFIX FIELD
OPECODE FIELD
EXTENSION FIELD
DST FIELD
RPTAMT FIELD
INSTRUCTION FETCH SECTION
130

FIG. 12

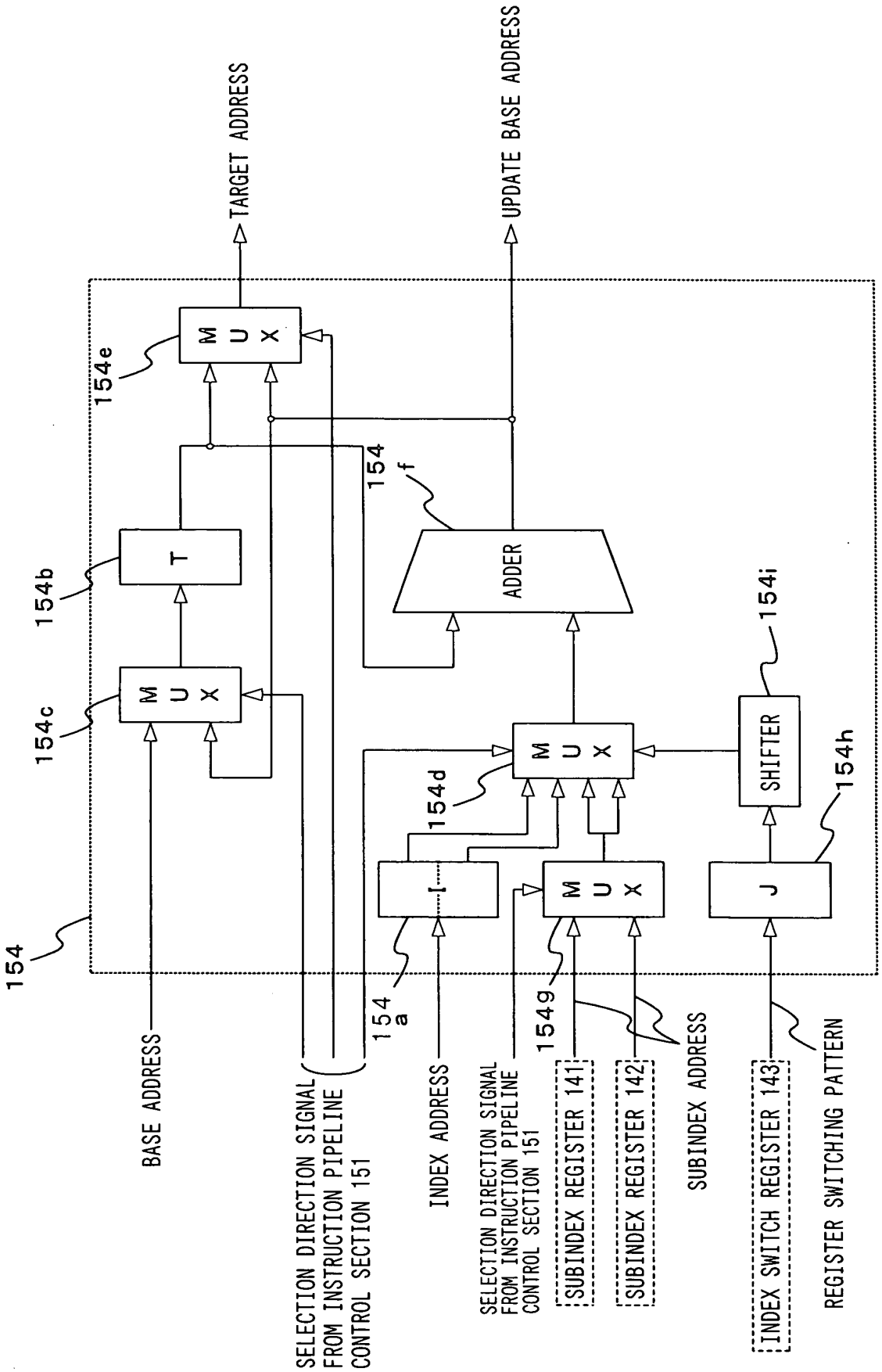

154
154a
154b
154c
154d
154e
154f
154g
154h
154i
T
I
J
MUX
MUX
MUX
MUX
ADDER
SHIFTER
TARGET ADDRESS
UPDATE BASE ADDRESS
BASE ADDRESS
SELECTION DIRECTION SIGNAL FROM INSTRUCTION PIPELINE CONTROL SECTION 151
INDEX ADDRESS
SELECTION DIRECTION SIGNAL FROM INSTRUCTION PIPELINE CONTROL SECTION 151
SUBINDEX REGISTER 141
SUBINDEX REGISTER 142
SUBINDEX ADDRESS
INDEX SWITCH REGISTER 143
REGISTER SWITCHING PATTERN

FIG. 13

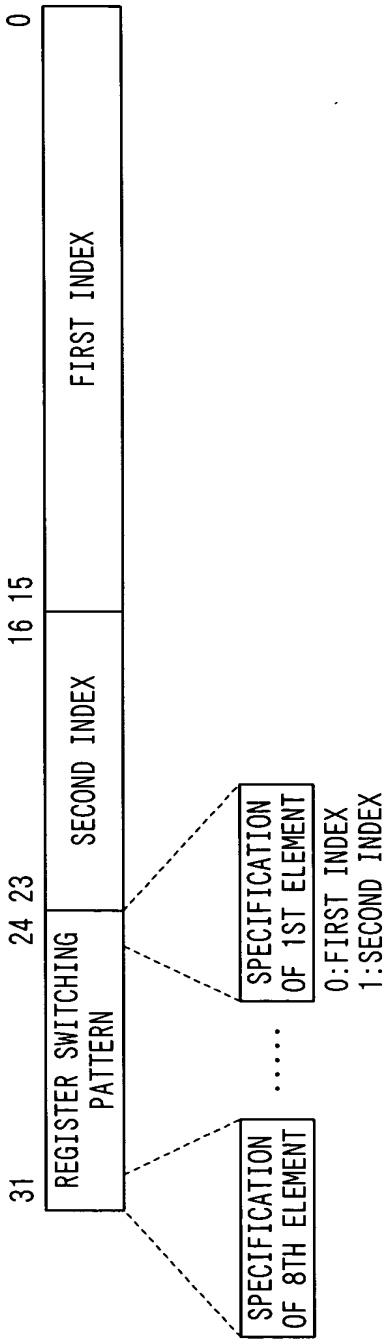
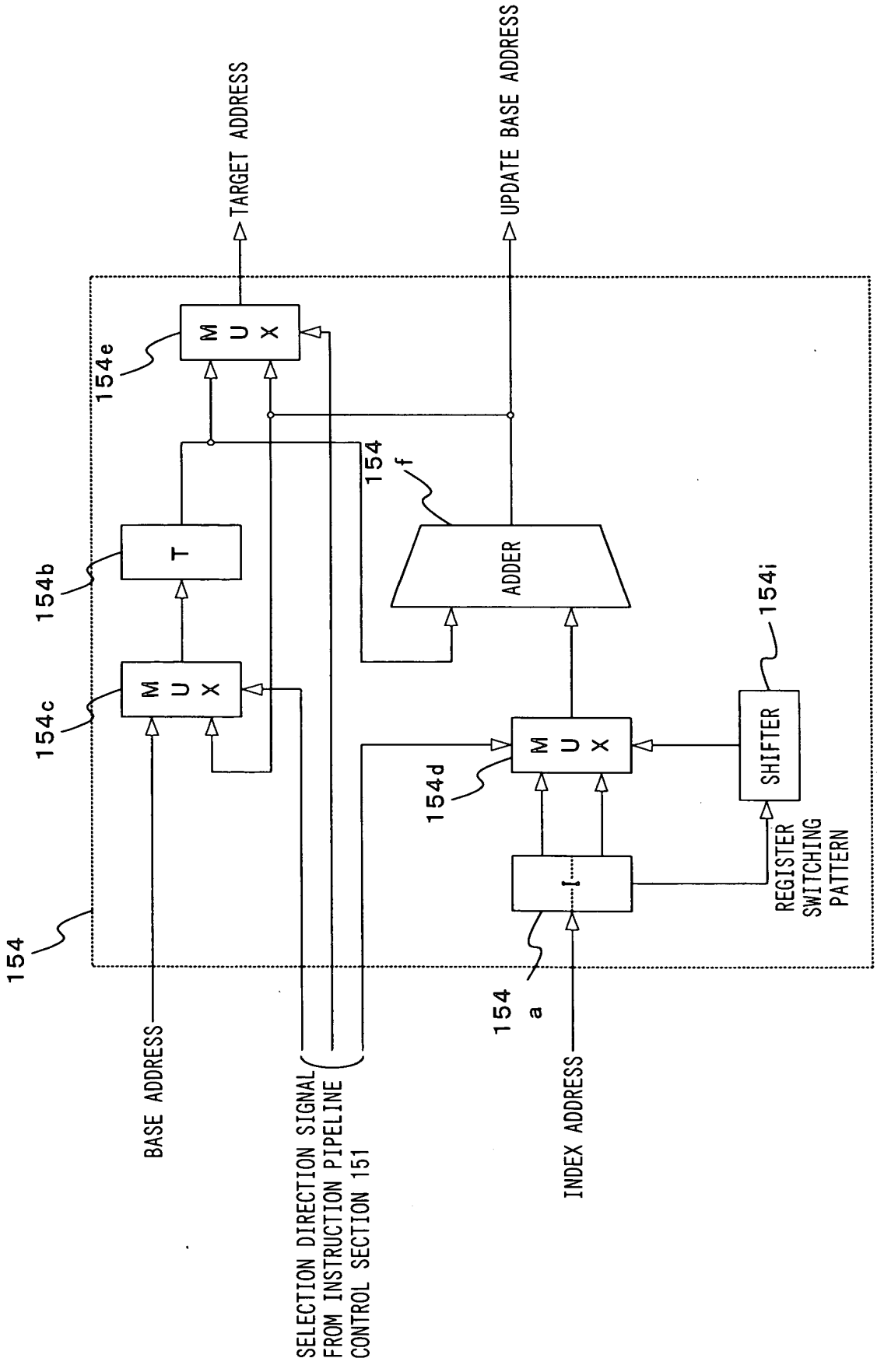

0
16 15
FIRST INDEX
SECOND INDEX
24 23
REGISTER SWITCHING PATTERN
31
SPECIFICATION OF 1ST ELEMENT
SPECIFICATION OF 8TH ELEMENT
.....
0:FIRST INDEX
1:SECOND INDEX

FIG. 14


TARGET ADDRESS
UPDATE BASE ADDRESS
154e
MUX
154
f
ADDER
154b
T
154c
MUX
154d
MUX
SHIFTER
154i
154
BASE ADDRESS
SELECTION DIRECTION SIGNAL
FROM INSTRUCTION PIPELINE
CONTROL SECTION 151
154
a
INDEX ADDRESS
REGISTER
SWITCHING
PATTERN